Artificial Life - Final Report

Afonso Esteves - fc 54394 e Miguel Dinis - fc 58198

Study the advantages and disadvantages of cyclic mutation and cyclic crossover in genetic algorithms applied to a simulation of a population of individuals.

Abstract

This study explores the application of cyclic mutation and cyclic crossover within genetic algorithms to optimize population survival in a simulated environment. Genetic algorithms, inspired by natural evolution, traditionally utilize standard mutation and crossover techniques, but these approaches often risk disrupting beneficial genetic structures. Cyclic variations aim to address this limitation by preserving genetic integrity while enhancing adaptability.

In this simulation, a population of virtual organisms, represented by neural networks, interacts in a closed environment where a contagious disease serves as a selection pressure. The genetic algorithms governing the population's evolution incorporate cyclic mutation and crossover strategies. Data from these approaches are compared against results obtained using standard methods, focusing on metrics such as genetic diversity, convergence rates, and overall population health.

Our findings highlight the potential advantages of cyclic strategies in maintaining advantageous traits, improving convergence speed, and optimizing adaptability. These results contribute to the understanding of cyclic genetic operations and their broader implications in evolutionary computation.

1. Introduction

Genetic algorithms (GAs) are optimization techniques inspired by the principles of natural selection and genetics. They are widely applied across domains such as machine learning, optimization problems, and simulation modeling due to their ability to find approximate solutions to complex problems. Basically, GAs rely on three main operations: selection, mutation and crossover.

Mutation plays a crucial role in introducing genetic diversity by altering genes in offspring, enabling exploration of the solution space. In standard approaches, mutation occurs randomly, which can disrupt beneficial genetic structures. Cyclic mutation, however, modifies genes in a predetermined sequence, reducing the likelihood of harmful changes while preserving advantageous traits. Similarly, crossover combines genetic material from two parent genomes to produce offspring. Cyclic crossover identifies and transfers gene cycles between parents to maintain structural integrity and enhance genetic performance.

This study builds on the foundational work of *Hoehn and Pettey* (1998), which explored parental and cyclic-rate mutation strategies in genetic algorithms, assessing their effects on genetic diversity and algorithm performance.

Among the variations of these operations, cyclic mutation and cyclic crossover have emerged as innovative approaches. Unlike standard methods, cyclic variations aim to preserve genetic structures and accelerate the learning process by minimizing the loss of beneficial traits. These methods cyclically modify subsets of genes or identify cycles during crossover to maintain key genetic information.

To explore this, we used four genetic algorithm variants, each with distinct mutation strategies:

a. *No-Parental, Low-Offspring Mutation* (NPLO): a standard genetic algorithm with no parental mutation and a low offspring mutation rate, serving as a baseline for comparisons.
b. *Low-Parental Low-Offspring Mutation* (LPLO): this algorithm introduces a low constant mutation rate at the parental level in addition to a low offspring mutation rate, mimicking chronic low mutation conditions.
c. *High-Parental Low-Offspring Mutation* (HPLO**)**: a high mutation rate is applied at the parental level, reflecting extreme environmental conditions or stressors, while maintaining a low offspring mutation rate.
d. *Cyclic-Parental Low-Offspring Mutation* (CPLO): mutation at the parental level alternates between low and high rates cyclically, mimicking acute periodic environmental shifts, combined with a low offspring mutation rate.

This study investigates the advantages and disadvantages of cyclic mutation and crossover in a simulated environment populated by virtual organisms. Each organism is represented by a neural network influenced by genetic encoding, with survival and reproduction driven by natural selection principles. A key challenge introduced in the simulation is a contagious disease, which tests the resilience and adaptability of the population under varying mutation and crossover strategies.

Through detailed simulation and comparative analysis, this project seeks to evaluate the efficacy of cyclic genetic operations

relative to standard methods, providing insights into their potential to optimize genetic algorithm performance.

2. The Article

The article by Hoehn and Pettey, titled *Parental and Cyclic-Rate Mutation in Genetic Algorithms: An Initial Investigation*, examines the effects of introducing mutation at both the parental and offspring levels in genetic algorithms. The authors propose a novel parental mutation genetic algorithm (GA) where mutation occurs not only in offspring but also at the parental level between evaluation and selection. Their experiments tested several variants, including standard constant-rate mutations and a cyclic-rate mutation strategy, across multiple optimization functions.

The study's findings highlight that cyclic mutation at the parental level can introduce genetic diversity while preserving advantageous traits, potentially improving convergence speed and robustness. Their work demonstrates that cyclic-rate mutation often outperforms constant-rate mutation strategies, offering a balance between genetic exploration and exploitation.

Our project builds on these findings by applying cyclic mutation and crossover strategies in a simulated population of virtual organisms, with a contagious disease serving as a selection pressure. While the article focuses on theoretical optimization tasks, our work extends this concept to a practical simulation scenario, evaluating how cyclic strategies affect survival, genetic diversity, and adaptability in dynamic environments. This enables a deeper understanding of the practical implications of cyclic genetic operations beyond abstract optimization tasks.

3. Our Approach

To evaluate the impact of different mutation strategies in genetic algorithms, this study employed a systematic methodology encompassing simulation design, algorithm implementation, and performance evaluation. Each approach was tested under controlled conditions using the following setup.

3.1. Simulation Environment

The simulation took place in a closed environment represented as a two-dimensional grid, where virtual organisms interact, move, and adapt. Each organism was governed by a simple neural network whose parameters were encoded in its genome. The neural network influenced the organism's decision-making, including avoiding infection, finding resources, or reproducing. A subset of the initial population was infected with a contagious disease, creating a selective pressure that favored the evolution of disease-resistant genes.

The simulation allowed organisms to interact with their environment for multiple iterations, equivalent to one generation. After each generation, the population's fitness was evaluated, and the genetic algorithm was applied to select the most suitable individuals for reproduction.

3.2. Project Structure

Our project comprises multiple interdependent components organized into modules within the source folder. The main modules include:

a. *Gen.py* - implements the core functions of the genetic algorithm, including mutation, crossover, and selection operations. These functions allow the genetic algorithm to evolve populations over generations by applying cyclic mutation and cyclic crossover strategies. Cyclic mutation modifies genes based on predefined patterns, preserving structural integrity while introducing variability. Similarly, cyclic crossover identifies gene cycles between parents, ensuring that offspring retain advantageous genetic traits. Selection is implemented through a tournament-based strategy, promoting diversity while maintaining competitive pressure.

b. *Simulation.py* - defines the closed environment where the organisms interact. The simulation grid provides spatial constraints, while functions in this module govern the spread of a contagious disease that serves as a selection pressure. The module also includes methods for evaluating the organisms' fitness based on their performance, which influences their likelihood of reproduction. Organisms with higher fitness are more likely to pass on their genetic information, guiding the evolution of the population.

c. *Organism.py* - defines each organism itself, represented by a simple neural network whose weights and biases are encoded in its genome. The neural network processes environmental inputs, enabling organisms to make decisions, such as navigating the grid or avoiding infection. The genome is initialized randomly and evolves through the genetic algorithm, with mutation and crossover operations modifying its parameters to optimize the network's performance.

d. *Main.py* - which serves as the entry point for the simulation. This script initializes the system, manages the simulation runs, and invokes the genetic algorithms. Each genetic algorithm variant—NPLO, LPLO, HPLO, and CPLO—is executed independently, with results collected for comparative analysis. It has a function that orchestrates the execution of each algorithm, iterating through generations and tracking key metrics such as best fitness and genome configuration. Results from the simulation are used to replay scenarios with the best-performing genomes, providing insights into the efficacy of each approach.

In comparison to the referenced study, our project extends the exploration of cyclic genetic operations by integrating them into a dynamic simulation of virtual organisms. While the article focuses on abstract optimization tasks, our implementation emphasizes real-world applicability, incorporating environmental factors and neural network-driven decision-making. By analyzing the outcomes under different mutation and crossover strategies, our project builds on the article's findings, demonstrating how cyclic genetic variations influence adaptability, diversity, and survival in complex systems.

3.3. Implementation of Mutation Strategies

The genetic algorithm was implemented to test four distinct approaches, corresponding to different combinations of parental and offspring mutation rates.

GA	**Parents**	**Offspring**
NPLO	N/A	0.1
LPLO	0.1	0.1
HPLO	0.8	0.1
CPLO	0.1, 0.8	0.1

Table 1: Mutation Rates for each Algorithm

These algorithms were executed sequentially using a shared framework. The *run* function made it easier to simulate each approach by setting the corresponding mutation parameters and running the genetic algorithm for a fixed number of generations.

3.4. Data collection and Analysis

Each algorithm was executed for 50 generations with a population size of 10 organisms. Fitness values were recorded at each generation to evaluate the algorithms' performance. Then we retrieved the best fitness values and genes of the most successful organisms, which were used to replay simulations with optimized populations. This enabled the comparison of how each mutation strategy influenced convergence rates, genetic diversity, and survival against the simulated disease. After the execution of our software the values are recorded from stdout in order to be analysed later by us. The algorithms run in the following order: LPLO, HPLO, CPLO, NPLO and this is the order used throughout the data analysis of our results. The data is then evaluated and discussed between multiple parties in order to understand whether it meets the expectations according to the information retrieved from the referenced article, how it relates to the conclusion achieved by other researchers and what can be concluded from this project, what useful information can be extrapolated.

4. Simulation Framework Elements

4.1. Key Operators for Evolution

A *tournament selection* strategy was employed. In this approach, subsets of the population are randomly sampled, and the fittest individuals from each subset are selected as parents. This method balances exploration and exploitation by introducing a degree of randomness while favoring fitter individuals.

The *mutation* was performed at the tensor level on the weights and biases of the neural network controlling each organism. Random perturbations are applied to these parameters with a specified mutation rate. Different mutation strategies (low, high, cyclic) were tested depending on the algorithm variant.

Crossover combines the genetic information of two parent organisms to produce offspring. For tensors, an element-wise crossover mask is generated, ensuring that each gene is inherited from one of the two parents. This preserves structural information while introducing genetic variation.

4.2. Sensors and Actions

Organisms in the simulation can rely on simple sensors and actions to interact with their environment.

The developed *sensor inputs* allow the organisms to detect grid positions by checking whether adjacent positions are occupied by other organisms, availability of movement in each cardinal direction and the sickness level of the near organism. All of them are processed by the organism's neural network to inform decision-making.

Based on neural network outputs, organisms can perform actions, such as moving to an adjacent grid cell in one of the four directions or even staying in their current position if no movement is deemed advantageous.

4.3. Simulation Parameters

Key parameters were carefully chosen to balance computational feasibility and meaningful results:

I. *Population Size* - Each simulation maintains a population of 50 organisms. This allows for sufficient genetic diversity while ensuring computational efficiency.
II. *Generations* - The genetic algorithm was run for 50 generations per simulation. This provides ample opportunity for evolutionary trends to emerge.
III. *Grid Dimensions* - The simulation environment is a 20x20 grid, providing a closed space for interaction and competition among organisms.
IV. *Maximum Steps* - Each simulation run lasts 20 steps, representing one complete lifecycle of the population.

4.4. Shared Genomic Structure

In this simulation, all organisms within the same simulation share a common genome. This choice simplifies the modeling of evolution by focusing on the collective adaptation of a single genetic structure rather than tracking individual genomes. This approach also emphasizes the genetic algorithm's ability to optimize global population traits rather than individual characteristics.

4.5. Sickness as a Fitness Metric

Sickness is a key factor in determining organism fitness. It is initialized randomly within a defined range (e.g., 30–60) and updated dynamically based on the organism's position on the grid. The lower the sickness value, the higher the fitness of the organism. Sickness serves as a selection pressure in the genetic algorithm, rewarding individuals or populations that can better navigate their environment or avoid detrimental conditions. This concept directly ties to the genetic algorithm by ensuring that

genes contributing to lower sickness levels are preserved and propagated in subsequent generations.

5. Project Results

We ran all the algorithms 3 times with the aforementioned parameters. At each generation the best individual is saved and when the GA concludes it presents us with the fittest individual it could come up with. According to the data recorded from said experiments each run of each GA got us the following fitness values: [80, 82, 80, 80], [91, 89, 78, 91] and [80, 83, 79, 80] with each tuple being matching the format: [LPLO, HPLO, CPLO, NPLO]. Additionally the convergence speed was also tracked. In order to measure the conversion speed we measured the number of generations required until the GA found an individual with fitness below a threshold T=100. These values are presented here [7,10,8,7], [8,14,8,6], [7,10,7,6] matching the values of the fitness presented earlier. It is important to remember that over 90% of the fitness values produced were in the number range]80,1600]. Finally it's possible to notice that in almost all GA that ran 50 generations no new best individual was found past generation 25-30 where all individuals were close to the 90 fitness value aside from the occasional bad mutation or crossover that would produce individuals with fitness > T again.

5.1. Data Analysis

From the data collected the first thing we can observe is that all methods reached a local maxima around generations 25-30 meaning that further generations didn't yield new best individuals and the remaining iterations were probably not necessary. Therefore we can observe that despite the approach of the GA the different applications of the mutation operator did not affect the speed at which a local maxima was found meaning that in terms of time all approaches will find their best answer relatively at the same time.

Next, taking a look at the convergence speeds of each approach we can notice that, although small, there were differences on the values between all algorithms. Noticeably, the more frequent mutation operations were applied to parents, the more it took for the GA to converge on smaller values. This difference despite being small and possibly related to the random initial state of the GA was reliably replicated throughout multiple tests leading us to believe that the application of mutation operations in parent individuals negatively impacts the convergence speed of the GA. However, while the convergence speed decreased, so did the fitness value of the best individual found by the GA when making use of parental mutation. It's noteworthy to point out that although the CPLO achieved the best possible individual in most of our experiments, it was still not last in convergence speed metric throughout our 3 study cases. From the observation we can derive the following conclusions: While increasing the mutation allowed the GA to better avoid local maxima and achieve better results, it often came with the price of reducing the convergence speed since more oftenly favorable genome was transformed and lost through mutations either in the parent or in the offspring, However CPLO manage to achieve the best results, comparable to HPLO, with a reduced penalty in the convergence speed. Therefore it's safe to assume that within the context of this experiment, if we can spare compromising some of the convergence speed we can get the best results with CPLO. This approach provides us with the best equilibrium of convergence speed vs best fitness maximizing the value from the GA.

5.2. Article Comparison

The findings from this study closely align with those presented in Hoehn and Pettey's research on parental and cyclic mutation in genetic algorithms. Both studies emphasize the advantages of cyclic mutation in maintaining genetic diversity and achieving superior fitness outcomes. CPLO emerged as the best-performing algorithm in both studies, consistently achieving optimal solutions by balancing exploration and exploitation.

One notable similarity is the impact of mutation rates on convergence speed and fitness values. Both analyses highlighted that high mutation rates can destabilize populations, leading to slower convergence and reduced overall performance. However, cyclic strategies were shown to mitigate these effects by periodically introducing diversity in a controlled manner. This allowed populations to escape local maxima and achieve better global solutions.

Despite these parallels, key differences exist between the two studies. Hoehn and Pettey's work focused on theoretical optimization tasks using benchmark functions, whereas this project explored practical applications in a dynamic simulation environment. The incorporation of environmental pressures, such as a contagious disease, added complexity and real-world relevance to the results.

Another distinction lies in the metrics used for evaluation. While the article relied on statistical analysis of benchmark function outcomes, this project employed fitness metrics tied to the survival and adaptability of virtual organisms. This shift in context demonstrates the versatility and applicability of cyclic mutation strategies across diverse scenarios.

To sum up, both studies reinforce the efficacy of cyclic mutation strategies in genetic algorithms. This project builds upon the foundational insights provided by Hoehn and Pettey by extending their applicability to complex, real-world-inspired environments. The results underscore the potential of cyclic genetic operations to enhance adaptability and optimize outcomes in dynamic systems

6. Conclusions

In conclusion, the experiment we ran showed promising applications for CPLO GA implementations where the problem requires an improved solution or has the tendency to be stuck on local maxima if it can spare a hit to its convergence speed, requiring more time to narrow down the population to fit individuals. Although our approach revealed data supporting the use of CPLO, the marginal differences it offers to traditional approaches could be the result of the small scope of the problem used to test the multiple GA's. Our sickness was measured in a crude and simple way and each organism within the simulation

had a very small set of sensors and actions. More studies evolving more complex environments could probably offer a more relevant insight on the applications of CPLO in real world problems with vastly superior degree of complexity. As it stands, this study offers a base case study for the viability of these approaches and notably parental mutation techniques in at least, small simple environments and corroborates some of the conclusions achieved by the authors of the referenced paper in their research on this same topic.

While the results demonstrated that CPLO can achieve higher fitness values and avoid local maxima, the differences observed between algorithms may reflect the limited complexity of the simulation. Factors such as the restricted sensory inputs, the homogeneity of the genome within simulations, and the small population size contribute to a narrower scope of evolutionary dynamics. These constraints, while necessary for computational feasibility, may obscure subtler distinctions between the algorithms or their performance in more dynamic and diverse environments.

Future research should aim to scale these experiments to larger, more complex systems. Expanding the range of sensors and actions available to organisms, introducing heterogeneity within populations, and simulating more intricate environmental pressures would provide a deeper understanding of how cyclic mutation strategies like CPLO perform in practical applications. By addressing these limitations, subsequent studies can build upon this foundational work to explore the full potential of cyclic genetic algorithms in solving complex, real-world optimization problems.

References

Journal Article

Theodore P. Hoehn and Chrisila C. Pettey.. Parental and Cyclic-Rate Mutation in Genetic Algorithms: An Initial Investigation.

This document was last revised on January 24, 2025.